# Speed Challenge: A Case For Hardware Implementation In Soft Computing

Taher Daud, Adrian Stoica, Tuan Duong, Didier Keymeulen, Ricardo Zebulum, Tyson Thomas, and Anil Thakoor
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, California, USA
taher.daud@jpl.nasa.gov, (818) 354-5782

## *Abstract*

*For over a decade, JPL has been actively involved in soft computing research on theory, architecture, applications, and electronics hardware. The driving force in all our research activities, in addition to the potential enabling technology promise, has been creation of a niche that imparts orders of magnitude speed advantage by implementation in parallel processing hardware with algorithms made especially suitable for hardware implementation. We review our work on neural networks, fuzzy logic, and evolvable hardware with selected application examples requiring real time response capabilities. Hardware designs include neural network chips performing object classifications for surface mappings and route determination, and high speed target recognition when neural ICs are stacked 64-thick as a sugar-cube, providing image correlation capabilities with nearly 1 tera-operations per second speed. Sensor data fusion requirement for multisensor scenario has led to development in fuzzy set hardware with three orders of magnitude speed advantage. Recently, genetic algorithm development for evolvable hardware systems implemented on suitable electronic hardware has shown very exciting high speed evolution of various digital and analog circuits as well as solution of T-norms and S-norms with fuzzy processing on a single test chip. The details of the hardware architecture and chip test results are described in this paper. Some of the details of processing are omitted; however, relevant references are provided.*

**Key Words:** Neural Networks, Fuzzy Logic, Evolvable Hardware, High Speed, Neuroprocessor, 3DANN.

## 1 INTRODUCTION

The Jet Propulsion Laboratory (JPL) is conducting research in various aspects of information processing. For the past fifteen years, JPL has been actively pursuing soft computing research involving theory, architecture, applications, and electronics hardware. The driving force in all our research activities has been two-fold: the potential enabling technology promise; and an innovation in performance that would provide orders of magnitude speed advantage by implementation in parallel-processing hardware. During early neural networks revival stage in the eighties, a building block synapse and neuron architecture was designed and tested [1-2]. This design with later improvements became the center-piece of a varied electronic circuitry, providing high data-processing speed and low power consumption at individual synapse and neuron levels, as well as for the overall subsystem. It provided a power-miser data processing circuit, which formed the basis for a multi-layer perceptron (MLP) architecture [3-4]. A more recent neural network design was used to stack a multi-chip module (64 identical chips) in a 3D configuration (sugar cube size) for high-speed image processing [5]. Further advancement of the architecture has been an innovation in algorithm towards on-chip learning, leading to 0.18μm feature size and silicon-on-insulator technology [6].

Later, computational intelligence techniques, such as fuzzy logic and neural networks combined with the more traditional Artificial Intelligence paradigm of expert systems proved efficient in solving a category of problems for which an accurate mathematical formulation of models was either not feasible or practically impossible to compute in useful time. Using these soft-computing paradigms, an architecture was evolved for sensor and data fusion, which was very effective in providing a solution in high-speed hardware. While these processors can be built both in digital or analog hardware, the massive amount of interconnection lines of a parallel implementation and the power requirements encountered in certain space, military or commercial applications such as hand-held devices make the idea of an analog ASIC processor preferable. An example of such an application requiring low power and fast processing of multi-sensor data is associated with the solution of object discrimination performed onboard a fast frame seeker.

The high speed processing used the analog hardware technology of neural networks, fuzzy logic, and expert rule with the conventional digital processing of a host computer. The individual modules were designed to be

reconfigurable and cascadable. In addition, the overall architecture was developed to be flexible enough for rerouting of signals to any required processing module by having an interconnecting network with switching arrays. We concentrate here on the fuzzy logic module. A new method of implementation of fuzzification resulted in a compact and low power design [7].

Present focus starting last year is for the new research activity in Evolvable Hardware (EHW). An appreciable amount of theoretical, algorithmic, modeling, and analytical work has been done. Further, a test chip has been fabricated, tested, and used successfully in various circuit evolution experiments [8]. Presently, genetic programming is being done in simulation; however, design work is ongoing to integrate it along with the evolvable circuits on a single chip.

A description of these activities along with the motivation brought on by required applications is described in brief. Selected results of the hardware implementation are presented. EHW work is described in greater details, again not only for the speed advantage but also its potential for system evolution leading to long-life mission-enabling nature.

# 2 AUTOMATIC TARGET RECOGNITION

Problems associated with automatic target recognition (ATR) have defied real time solutions for decades. Despite recent progress in microprocessor technology, deployable systems to-date are either bulky, power hungry, or not capable of providing reliable target recognition and tracking in real time. Parallel computer systems capable of giga-operations/s, such as Adaptive Solutions' CNAPS array processors could perform convolution with a small kernel (3x3 or 8x8) in real time. However, achieving general object recognition from video in real time with a reasonable template size (32x32 or larger) is still beyond today's computer/processor technology [9]. Optical correlators designed for distortion-invariant pattern recognition are an attractive alternative for achieving massively parallel processing with photons [10]. However, they have yet to overcome many performance and system issues, such as signal-to-noise, discrimination ability, programmability and limitations of available spatial light modulators, post-processing of correlation outputs, and packaging. This prevents the realization of a flexible, robust optical processor for ATR.

## 2.1 ATR Testbed

Our approach is based on 3D IC chip architecture that is capable of processing any size image (whether IR, UV, or visible) by sequentially inputting consecutive 64x64 windows and performing high speed convolutions with 64 prestored 64x64 image templates. The data processing architecture, which provides an image processing system in a small package and is suitable for ATR was designed and executed as a testbed [11]. This integrated testbed incorporates an advanced artificial neural network processor and several new sensors covering the spectral range from ultraviolet (UV) through infrared (IR). The convolution inner-product outputs are fed to a 64-input neural network trained to recognize and classify objects (number of NN outputs depend on number of objects to be classified).

## 2.2 Analog Neural Networks

On the image data-processing side, the neural network advances have led to chip implementations that have been assembled into innovative, three-dimensional architectures capable of data processing speeds with $10^{12}$ operations per second (OPS) and running 64 image-based convolutions (with 64x64 kernel size) in real time. 3D Artificial Neural Network (3DANN) is a sugar-cube-sized, low-power neuroprocessor with its IC stack [12-13] with its 64 outputs coupled to a 64-input neural network, presently implemented using a SHARC parallel processor. The main convolution processor is a fully testable 64-chip cube that allows processing of any size image by acquisition and working on a 64x64 window at a time.

The main contribution for the testbed has been the convolution engine as the neuroprocessor, particularly designed to aid in recognition of shapes in resolved images at extremely high speeds ($10^{12}$ OPS). As mentioned earlier, 3DANN uses a synapse design based on Multiplying Digital-to-Analog Converter (MDAC) technology using a hybrid approach reported in detail in Refs. [2,4]. Each circuit is digitally programmable, has an 8-bit resolution digital weight storage, and is an analog multiplier with a voltage-input/current-output configuration [14]. In it, 64 complete inner products, each with a 4096 (i.e., 64x64) input array can be accomplished in 250 nanoseconds (i.e., $\sim 10^{12}$ multiply and add operations in 1 second. As a comparison, the convolution operations on a 256x256 image would take 16 milliseconds using 3DANN. However, the same processing would take about 2.5 hours using

SPARC-10, and a few minutes with parallel processor such as SHARC. As can be seen, the hardware provides quite a few orders of magnitude speed advantage.

### 2.2.1 Building Block Chips

We first describe the MDAC design for the synapse here. The designs have been geared towards low power and compact implementations. Analog current summation of multiple synapses can easily be done on a connection wire and, if required, can be fed as input to a neuron. Further, the neurons provide an output as a voltage signal which, with multiple fan-outs, can be distributed in a fully parallel fashion to a number of synapses through connection wires.

### 2.2.2 Synapse Chip Design

Our synapse chip design is a 64x64 synaptic crossbar matrix with 64 input and 64 output lines. A synapse is placed at each node of the matrix. For ease of weight downloading using a host computer, a digital scheme has been used for weight storage. The synapse design is based on a static random access memory (SRAM) with 7 bits (6 bits + sign bit) of resolution having two-quadrant current multipliers as DACs. In Figure 1, the circuit containing the digital switches D0 to D5 represent the MDAC. Decoders for row/column selection and address/data lines are included for random accessibility and programmability as shown within the inset, which determine the weight $W_{ij}$. Input signal is fed as analog voltage $V_{in}$, which is converted as an equivalent current $I_{in}$ and fed through current mirrors to all the synapses along that row. Each synapse then multiplies $I_{in}$ with its own digitally stored multibit weight, $W_{ij}$. As shown in Figure 1, the weight multiplication is obtained through one or more of the six parallel

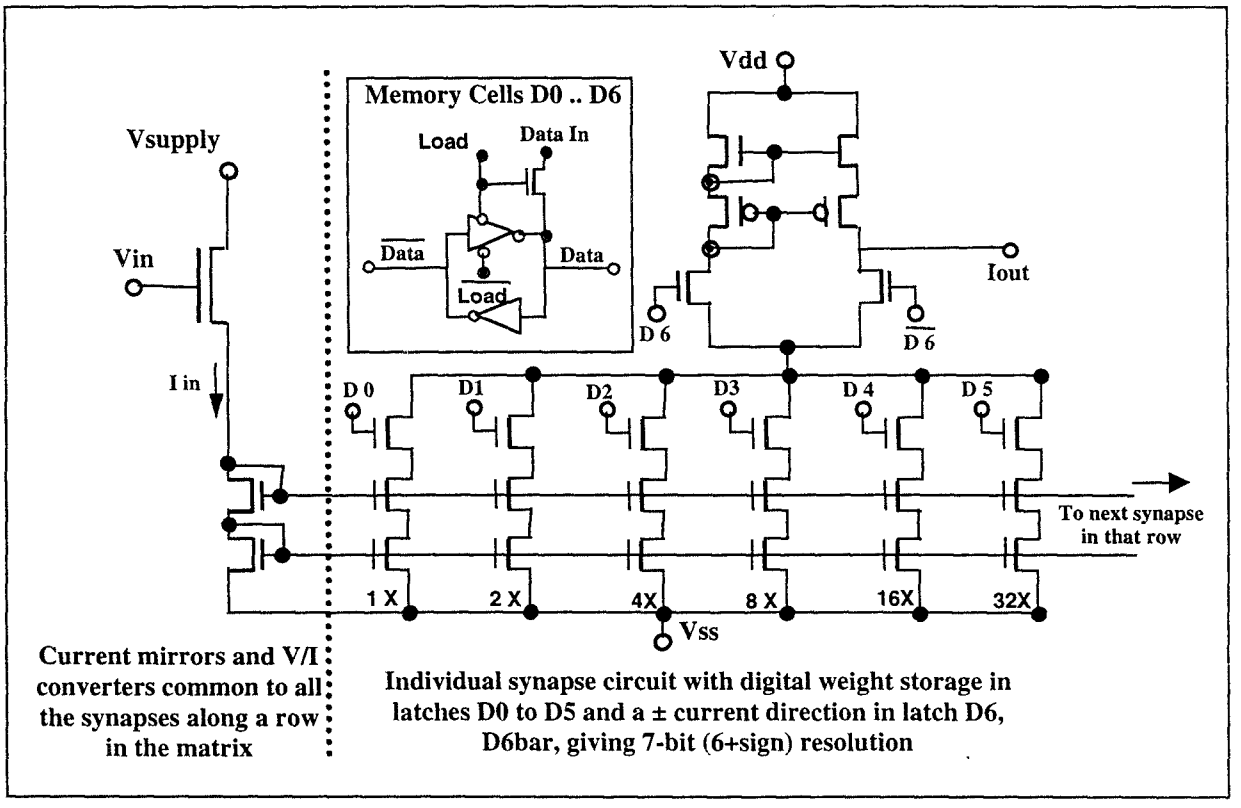


Figure 1. Synapse circuit containing voltage to current input stage, multi-bit Multiplying Digital to Analog Converter (MDAC), and a multi-bit digital memory.

stages of cascode current mirrors by selective operation of the digital latches D0 through D5 already performed during the previous weight storage operation. By using the upper current steering circuit and by programming of the latch (D6 - D6bar), the sign bit is obtained with a positive or negative current flow as sinked or sourced.

Similarly, an 8-bit synapse would require an additional bit with a set of cascode stages adding to the digital logic as well. To conserve on space, a slight modification was used by splitting the digital bits into a coarse and a fine set of cascode columns with two sets of current mirrors. The input current to one current mirror was Iin and that of the other was made 64*Iin by suitably altering the design of one of the two input transistors. However, the same voltage Vin was fed at both inputs.

## 2.3 Functional Description Of Analog Processing

The 64 analog voltage inputs first get converted to currents by a set of V-I converters at the beginning of each row of the 64 x 64 synaptic array. These signals are then current mirrored into all 64 synapses along the row so that all the synapses in a given row receive an identical input [15].

A byte, which controls switches to scale current copies of the input, is stored in a local static memory (SRAM) for each synapse. By switching in different multiples of the input current and adding them together, the input current is effectively multiplied by the digital weight stored in the local SRAM. Synapses on the same column have their outputs, Iout, summed by attaching them all to the same wire. These 64 summed signals, one for each column of the array, are then sent directly out every 250 nanoseconds as shown in Figure 2 [13-14]. In the 3D scheme, respective column outputs of all the 64 chips are added together through edge-wise metallization, so that a 64x64 incoming image multiplies with 64, 64x64 templates on 64 chips to provide 64 analog inner-product current outputs. The process is repeated by consecutively inputting the adjoining 64x64 image windows, "row-by-row" and "column-by-column". Thus a 256x256-image convolution would be completed in about 16 milliseconds.

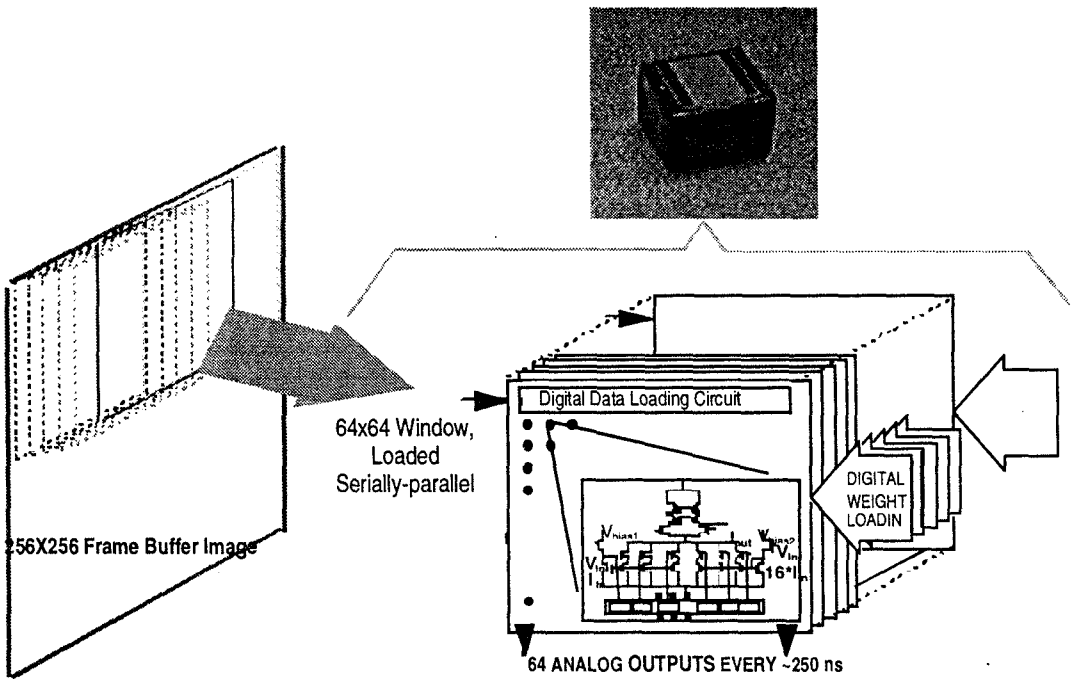


Figure 2. The 3DANN consists of 64 layers of a 64x64-synapse array based on an 8-bit MDAC. It incorporates a special-purpose image-write circuitry. 3D network is realized (as pictured) in a 10-gm, 3-cm$^3$ package, with power consumption of ~2.5W. The image loading is specially designed for rastering of a 64x64 window of a larger image in the frame buffer and is synchronized with 3DANN's 250-ns inner-product operations.

## 2.4 Digital Weight Programming:

Before the processing can begin, the synapse weights as 64x64templates are obtained separately. For this purpose, an approach based on eigenvectors is employed. Since each data point (image) is a 4096-element vector, finding a set of 4096 orthonormal eigenvectors is possible (64 of these can then reside on 3DANN at a time). Selecting the most significant 64 eigenvectors constructed from principal component analysis of target imagery reduces dimensionality of image sets, without losing much of the information relevant for classification. The selected 64 templates are loaded single row at a time. The data for a given row is clocked into a 64 long, 8-bit wide shift register, one byte at a time. After 64 clock cycles, the data for an entire row of synapses is ready to be loaded into the local memory of each MDAC. A 6-bit row address is supplied and an active-low load signal is asserted, which dumps the data into the synapses on the row specified. The register loads from the bottom up so that the first data loaded corresponds to the first row. More details including its configuration as a recurrent neural network can be found in the literature [4,15].

Incorporating a multi-synapse circuit as an analog multiplier makes the network extremely powerful image-processing engine capable of carrying out in parallel 64 convolutions of the form:

$$\mathbf{c}_i(x,y) = \mathbf{f}(x,y) \otimes \mathbf{g}_i(x,y);\ i = 1, 2, ..., 64; \qquad (1)$$

where $\mathbf{f}$ is the input image, $\mathbf{g}_i$ is the filter mask, and $\mathbf{c}_i$ is the output image [11].

## 2.5 Processing Results:

Information about the object (its class, identity, or pose) is processed in a coarse-to-fine manner. For instance, after detecting an object in a frame, a rough estimate of image pose/scale is made, a result that can then be used to limit the variation that needs to be considered during object classification (i.e., plane, helicopter, and missile). Results (Figure 3) using the technique described here have achieved nearly 97% detection rates, 94% classification rates for determining the angle of the principle dimension of an object with respect to the image (±30°), and object classification rates approaching 95%. Results on object/non-object image classification rates achieved with a helicopter/missile/plane data set were also very encouraging [11].

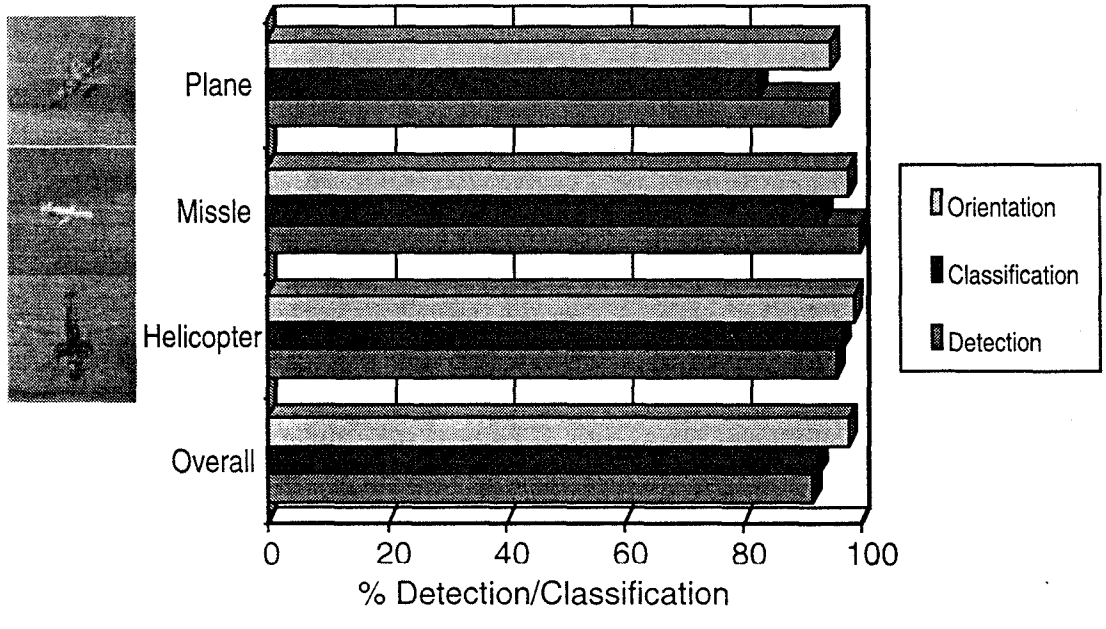


Figure 3. High detection/classification rates are achieved on selected data sets that include all possible orientations and scales of targets.

# 3 SENSOR FUSION PROCESSOR

The Sensor Fusion Processor (SFP) was developed to seamlessly combine rule-based systems, fuzzy logic, and neural networks to achieve parallel fusion of sensor data in compact low power VLSI. The first demonstration of

the concept was for the sensor fusion functionality; other applications, mainly in robotics and autonomous systems were considered for the future. The main assumption behind SFP is that fuzzy, rule-based and neural forms of computation can serve as the main primitives of an "intelligent" processor. Thus, in the same way classic processors are designed to optimize the hardware implementation of a set of fundamental operations, SFP was developed as an efficient implementation of computational intelligence primitives, and relied on a set of fuzzy set, fuzzy inference and neural modules, built in programmable analog hardware. The hardware programmability allowed the processor to reconfigure into different topologies, taking the most efficient hardware implementation during each phase of information processing. Following software demonstrations on several sets of data, three important SFP building blocks (a fuzzy set preprocessor, a rule-based fuzzy system and a neural network) were fabricated in analog VLSI hardware, which demonstrated microsecond level of processing times. The neural network module was designed on the same basis as described earlier, except that the same chip also contained an amplifier as a neuron with nearly sigmoidal transfer function. Here, we only describe the hardware details of the fuzzy set processor as an example of high speed processing. More details about the whole processor have already been presented in Ref. [7,16].

## 3.1 The Fuzzy Set Processor (FSP)

The main function of a fuzzy set processor is signal transformation, which can be interpreted, for example, as:

- fuzzification - i.e. association between an input crisp signal and a degree of membership to a fuzzy set/class; or
- signal conditioning/ non-linear transformation, coordinate transformation.

The FSP was designed as a processing module with 16 inputs of 5 membership classes each. The architecture of the FSP is presented in Figure 4. The chip has 16 analog voltage inputs and 16x5 outputs, and allows digital

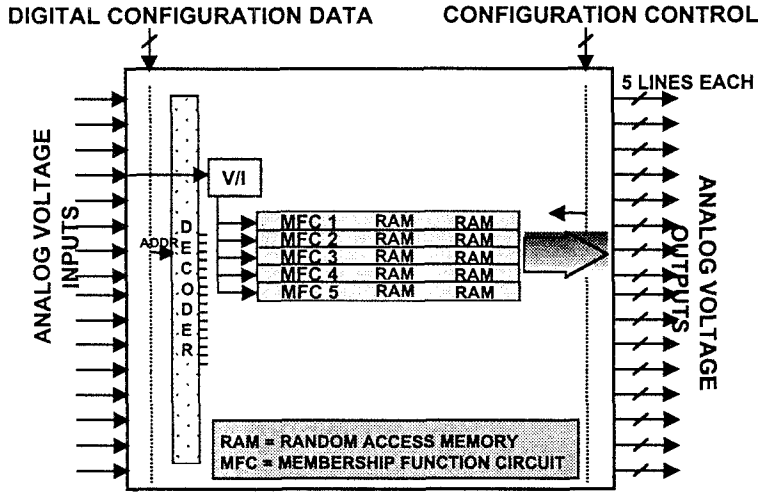


Figure 4. Fuzzy Set Processor (FSP) architecture

programmability of the membership functions for each input variable. The membership functions have trapezoidal shape, with programmable parameters for the legs and slopes as illustrated in Figure 5 inset. The position of the legs can be specified with 8-bit resolution and the slope with 5-bit resolution. The equations that describe the output of a trapezoidal membership function are:

| | |
|---|---|
| If $X <= A$, | then Y = Low |
| If $A < X = \text{or} < (CD+AB)/(B+C)$, | then Y = MIN(BX-AB + Low, High) |
| If $(CD+AB)/(B+C) < X < D$, | then Y = MIN(-CX + CD + Low, High) |
| If X> or = D, | then Y= Low, |

where A is the location of the left leg, B is the unsigned slope of the left leg, C is the unsigned slope of the right leg, and D is the location of the right leg. The chip design currently uses Low = 1 volt and High = 4 volts with Vdd = 5 volts.

The schematic diagram in Figure 5 details the processing path of a single membership function circuit (MFC). While inputs and outputs are in voltage mode for external compatibility, the internal MFC implementation is in

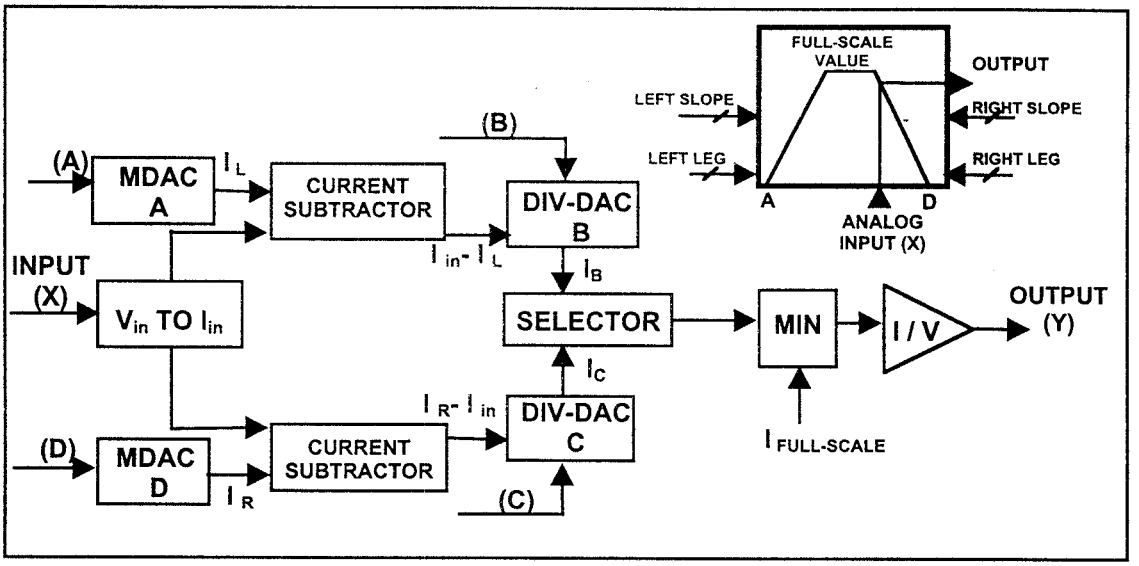


Figure 5. Block diagram of HW implementation for a simple MFC

current-mode. The input voltage enters the first processing block, which is a Voltage to Current (V/I) converter. Currents proportional to the digital values of the legs, A and D, are generated in Multiplying Digital to Analog Converters (MDACs). The current corresponding to the left leg gets subtracted from a copy of the input current, while a different copy of the input current gets subtracted from the right leg current. The resulting currents, which correspond to the left and right sides of the trapezoid, enter their appropriate Dividing Digital to Analog Converter (divDAC) where the signals are divided by 5-bit digital values to scale the slopes. The minimum of the two resulting values is then selected which chooses the side that is along the trapezoid. The top of the trapezoid is achieved by taking the minimum of the resulting current and the full-scale current, and this result is converted to the voltage output of the MFC. A test chip for 2 input variables with 5 membership functions calculating the degree of membership has been implemented and tested. A variety of membership functions generated by the chip are illustrated in Figure 6 [16-17].

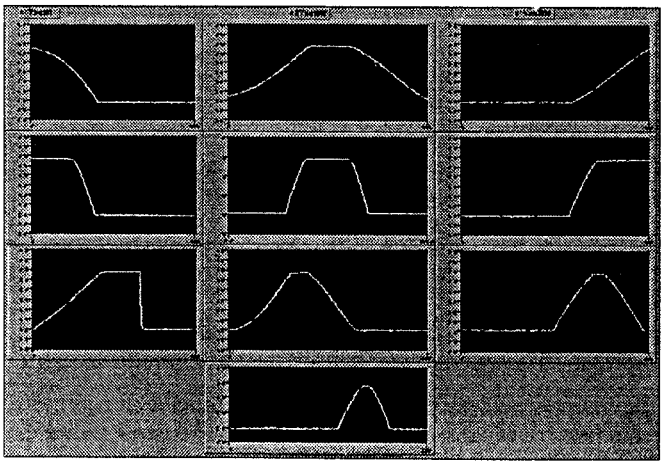

Figure 6. A variety of membership function shapes generated on the MFC test chip

The chip was used in an object discrimination/classification task and the results were compared with the software implementation for accurate reproduction in hardware of the results obtained by simulation. The membership functions were used to separate the spaces containing target objects and nonobjects. The software results showed that signals processed using these membership functions resulted in discrimination of objects and nonobjects, as well as discrimination of objects of different types, based on available data. The hardware tests showed that the fuzzification/ discrimination of this type takes less than a microsecond as compared to milliseconds to seconds for software simulations. Thus, this test showed that at least three orders of magnitude speed advantage was obtained by hardware over simulation. Further, if the problem size increases, the hardware takes the same time in parallel processing, whereas the time of sequential computers in simulations increases.

# 4 EVOLVABLE HARDWARE

The application of evolution-inspired formalisms to hardware design and self-configuration lead to the concept of evolvable hardware (EHW). In the narrow sense, EHW refers to self-reconfiguration of electronic hardware by evolutionary/genetic reconfiguration mechanisms. In a broader sense, EHW refers to various forms of hardware from sensors and antennas to complete evolvable space systems that could adapt to changing environments and, moreover, increase their performance during their operational lifetime.

We describe evolution-oriented devices and an evolvable system on a chip. A Field Programmable Transistor Array (FPTA) architecture is used as the experimental platform for evolutionary experiments. The platform is quite flexible and supports implementation of both analog and digital circuits. While previous works [8,18-19] illustrated the implementation of several conventional building blocks for electronic circuits such as logical gates, transconductance amplifiers, filters, Gaussian neurons, etc., here we describe an automatic design of the rather more unconventional circuits for combinatorial fuzzy logics.

In this part of our paper, we first present the components of an evolvable hardware system, and survey some important evolutionary experiments and applications of evolvable hardware. We also describe an evolution-oriented architecture based on the concept of FPTA, leading to an illustration of how the FPTA can be used to evolve reconfigurable circuits for combinatorial fuzzy logic. Circuits implementing parametric triangular norms are evolved in software and in hardware directly on the chip.

## 4.1 Evolutionary Synthesis Of Electronics

The main idea of evolutionary/genetic algorithms is inspired by the principle of natural selection. In nature the fittest individuals survive and reproduce passing along their genetic material to their offspring, who will inherit the characteristics that made the parents successful. Similarly, the evolution of artificial systems is based on a population of competing designs, the best ones (i.e. the ones that come closer to meeting the design specifications) being selected for further investigation. The offspring of this elite, in which pairs of parents were randomly selected for "mating", combine genetic material from two parents and may suffer genetic "mutations" (alternatively, in asexual reproduction the genetic code from one successful individual may be inherited, possibly with some random mutation). The offspring are new generation of competing designs. The process of trial-and-error parallel search can last many generations, and be constructed with many choices on how to implement reproduction, selection, etc.

The concept of evolvable hardware was born partially inspired by search/optimization/adaptation mechanisms and partially by the possibility of a design of reconfigurable devices such as Field Programmable Gate Arrays (FPGA). Circuits can be evolved by reconfiguring programmable devices (this is termed *intrinsic* EHW) or by evolution using software models – descriptions of the electronic H/W (referred to as *extrinsic* EHW).

Figure 7 illustrates the main steps of evolutionary design for electronic circuits. Each candidate circuit design is associated with a "genetic code" or chromosome. The simplest representation of a chromosome is a binary string, a succession of 0s and 1s that encode a circuit. The first step of evolutionary synthesis is to generate a random population of chromosomes. The chromosomes are then converted into a model that gets simulated (e.g. by a circuit simulator such as SPICE) and produces responses that are compared against specifications.

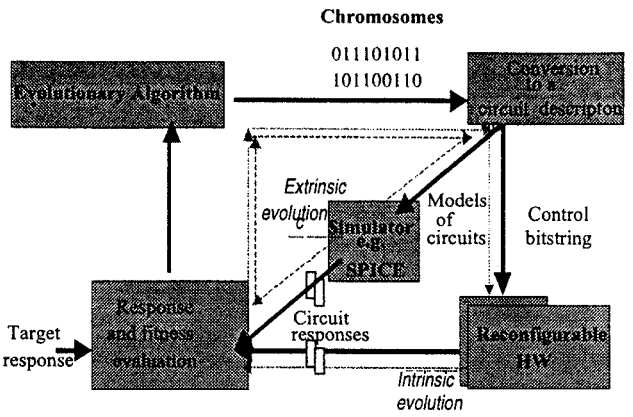


Figure 7. Evolutionary synthesis of electronic circuits

A solution determined by extrinsic evolution may eventually be downloaded or become blueprint for hardware. In intrinsic evolution the chromosomes are converted into control bitstrings, which are downloaded to program the reconfigurable device. The configuration bitstring determines the functionality of the cells of the programmable device and the interconnection pattern between cells. Circuit responses are compared against specifications of a target response and individuals are ranked based on how close they come to satisfying it. Preparation for a new iteration loop involves generation of a new population of individuals from the pool of the best individuals in the previous generation. Here, some individuals are taken as they were and some are modified by genetic operators, such as crossover and mutation. The process is repeated for a number of generations, resulting in increasingly better individuals. The process is usually ended after a given number of generations, or when the closeness to the target response has been reached. In practice, one or several solutions may be found among the individuals of the last generation.

## 4.2 Evolutionary Experiments

A variety of circuits have been synthesized through evolution and reported in literature [20-23]. Evolutions of analog circuits reported in [20] were performed in simulations, without concern for a physical implementation, but rather as a proof-of-concept to show that evolution can lead to designs that compete with human designs, or even exceed them in performance.

Current programmable analog devices are very limited in capabilities and do not support the implementation of the designs resulted in simulations (but, in principle, one can test their validity in circuits built from discrete components, or in an ASIC). More recently, intrinsic evolutionary experiments were performed on commercial Field Programmable Analog Arrays (FPAA), custom-designed ASIC as well as other devices.

Obtaining circuit designs through evolution requires many evaluations. Since analog programmable devices, flexible enough for the purpose of EHW are not available commercially, researchers have one of two options: Use simulations or build their own devices that can evolve. SPICE simulations have to solve differential equations and scale badly with increase in the number of nodes in a circuit. For example, a transient analysis simulation for a circuit with ~100 transistors takes about 1 second on a SUN computer.

We chose the second option and built silicon devices for evolution experiments using electronic circuits. To illustrate the speed advantage, the same circuit mentioned above that takes 1 second for simulation on SUN computer responds in ~100 ns when evolving hardware is used. Depending on the data acquisition system, including the speed of the a/d converters for acquiring a sufficient number of samples, we can estimate ~10μs to 1 ms as the evolution response time. Note, however, that this H/W approach is size independent and the same time would be required even for a circuit 100 times bigger while in software it would require ~10,000 times more time.

## 4.3 Building An Evolvable System-On-A-Chip

The efforts toward hardware evolution have been limited to simple circuits. In particular, for analog circuits, this limitation comes from a lack of appropriate reconfigurable analog devices to support the search, which precludes

searches directly in hardware and requires evolving in software on hardware device models. Such models require evaluation with circuit simulators such as SPICE; the simulators need to solve differential equations and, for anything beyond simple circuits, they require too much time for practical searches of millions of circuit solutions. A hardware implementation is expected to offer a substantial advantage in circuit evaluation time; in certain cases the time for hardware evaluation could be seconds instead of days as in case of evaluation in software.

For efficiency of EHW applications, future reconfigurable devices would benefit from *implementing evolution-oriented* reconfigurable architectures (EORA). One of the most important features for EORA relates to the *granularity* of the programmable chip. FPAA offer only coarse granularity, which is a clear limitation; FPGAs are offered both in versions with coarse grained and fine-grained architectures (going to gate level as the lowest level of granularity). From the EHW perspective, it is interesting to have *programmable granularity*, allowing the sampling of novel architectures together with the possibility of implementing standard ones. The optimal choice of elementary block type and granularity is task dependent. At least for experimental work in EHW, it appears a good choice to build reconfigurable hardware based on elements of the lowest level of granularity. Virtual higher-level building blocks can be considered by imposing programming constraints. Ideally, the "virtual blocks" for evolution should be automatically defined/clustered during evolution. In addition EORA should be *transparent architectures*, allowing the analysis and simulation of the evolved circuits. They should also be robust enough not to be *damaged* by any configuration existent in the search space, potentially sampled by evolution. Finally, EORA should allow evolution of both analog and digital circuits.

An evolvable system-on-a-chip architecture is suggested in Figure 8. The main components are a Field Programmable Transistor Array and a Genetic Processor. The idea of a field programmable transistor array was introduced as a first step toward EORA [24]. The FPTA is a concept design for hardware reconfigurable at transistor level. As both analog and digital CMOS circuits ultimately rely on functions implemented with transistors, the FPTA appears as a versatile platform for the synthesis of both analog and digital (and mixed-signal) circuits. The architecture is cellular, and has similarities with other cellular architectures as encountered in FPGAs (e.g. Xilinx X6200 family) or cellular neural networks. One key distinguishing characteristic relates to the definition of the elementary cell. The architecture is largely a "sea of transistors" with interconnections implemented by other transistors acting as signal passing devices (gray-level switches), and with islands of RC resources in-between.

Figure 9 illustrates an FPTA cell consisting of 8 transistors and 24 programmable switches. The status of the switches (ON or OFF) determines a circuit topology and consequently a specific response. Thus, the topology can be

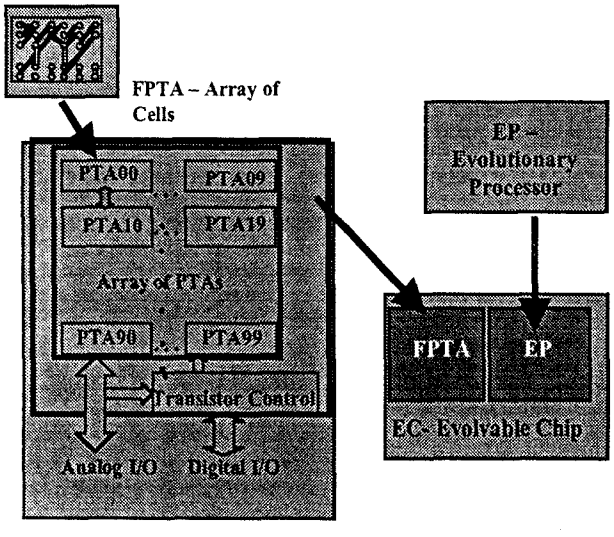


Figure 8. Schematic of an evolvable System-On-Chip.

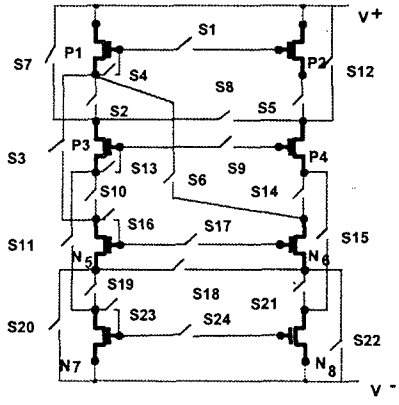


Figure 9. Module of the Field Programmable Transistor Array.

considered as a function of switch states, and can be represented by a binary sequence, such as "1011..." where by convention one can assign 1 to a switch turned ON and 0 to a switch turned OFF. Programming the switches ON and OFF defines a circuit for which the effects of non-zero, finite impedance of the switches can be neglected in the first approximation (for low frequency circuits).

## 4.4 Evolving Reconfigurable Circuits For Fuzzy Logic

This section illustrates the evolutionary design of infinitesimal multi-valued logic circuits, more precisely circuits for fuzzy logic. The objective is to determine circuit implementations for conjunctions and disjunctions for fuzzy logic. In such logic, conjunction and disjunction are usually interpreted by a *T-norm* and by its dual *T-conorm* (*S-norm*) respectively. A function **T**: [0,1] x [0,1] => [0,1] is called a triangular norm (T-norm for short) if it satisfies the following conditions:

- associativity ($\mathbf{T}(x,\mathbf{T}(y,z)) = \mathbf{T}(\mathbf{T}(x,y),z)$),
- commutativity ($\mathbf{T}(x,y) = \mathbf{T}(y,x)$),
- monotonicity ($\mathbf{T}(x,y) \leq \mathbf{T}(x,z)$, whenever $y \leq z$), and
- boundary condition ($\mathbf{T}(x,1) = x$).

A function **S**: [0,1] x [0,1] => [0,1] is called a triangular conorm (T-conorm or S-norm for short) if it satisfies the conditions of associativity, commutativity, monotonicity, and the boundary condition $\mathbf{S}(x, 0) = x$. **S** and **T** are corresponding (or pairs) if they comply with De Morgan's laws. Frank's parametric T-norms and T-conorms (also referred to as fundamental T-norms/conorms in [25]) were the selected choice for modeling the logical connectives. The family of Frank T-norms is given by

$$T_s(x,y) = \begin{cases} MIN(x,y) & \text{if } (s = 0) \\ x \cdot y & \text{if } (s = 1) \\ \log_s\left[1 + \dfrac{(s^x - 1).(s^y - 1)}{s - 1}\right] & \text{if } (0 < s < \infty),\ s \neq 1 \\ MAX(0, x + y - 1) & \text{if } (s = \infty) \end{cases} \quad (1)$$

The family of Frank T-conorms is given by

$$S_s(x,y) = \begin{cases} MAX(x,y) & \text{if } (s = 0) \\ x + y - x.y & \text{if } (s = 1) \\ 1 - \log_s\left[1 + \dfrac{(s^{1-x} - 1).(s^{1-y} - 1)}{s - 1}\right] & \text{if } ((0 < s < \infty),\ s \neq 1) \\ MIN(1, x + y) & \text{if } (s = \infty) \end{cases} \quad (2)$$

Electronic circuits implementing the above equations can be used in implementations of fuzzy logic computations or in implementing fuzzy S-T-norm neurons. One interesting application made possible in this implementation is the selection of the most appropriate s-parameter for the application at hand. Examples of the influence of various T-norms and S-norms in fuzzy control and automated reasoning applications can be found in [26-27] and for learning in fuzzy neurons in [28].

## 4.5 Evolution Results

The following preliminary results illustrate the possibility of evolving circuits that implement T and S for various values of the parameter *s*. The circuits were powered at 5V and the signal excursion was chosen between 1V (for logical level "0") and 4V (for logical level "1"). Intermediary values were in linear correspondence; i.e. 2.5V corresponds to logic level 0.5, etc.

The experiments were performed both in software (Spice simulations, extrinsic evolution) and in hardware (intrinsic evolution) using either one or two FPTA cells on a chip. The experiments used a population size of 128 individuals, were performed for 400 generations (with uniform crossover, 70% crossover rate, 4% mutation rate, tournament

selection) and took around 15 minutes using 16 processors when evolving in simulations. Each switch in FPTA cell has an associated control bit in a direct mapping. Thus there are 24 bits in the chromosome describing one cell. Interconnection experiments were done mostly with 4 bits. Thus a 2 cell experiment would use 52bits (24*2+4).

Figure 10, as an example, shows the response of circuits targeting the implementation of fundamental T-norms for s=100. The diamond symbol (◊) marks points of simulated/measured response of evolved circuit, while the cross symbol (+) marks the points of an ideal/target response for the given inputs. The output (T) is mapped on the vertical axis; values on axis are in Volts. The circuit responses for other s values such as s=0 and s=10 were similar. The circuits for T- & S-norm with s=100 were mapped on two FPTA cells. Figure 11 shows the diagonal cut for the same S-norm. All these responses were for circuits evolved in software.

For comparison, the response of a circuit evolved in hardware (for s=100) is shown in Figure 12, whereas the convergence toward solution can be seen in Figure 13. Sometimes the actual response (Figure 12) has a higher

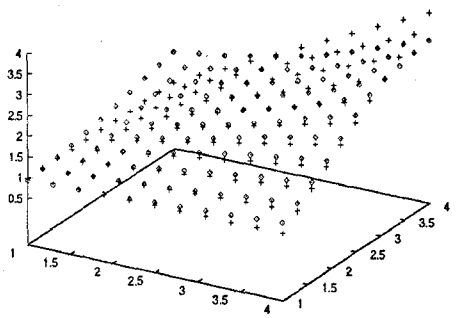

Figure 10. Response of a circuit implementing the fundamental T-norm for s=100 (◊). Target characteristics are shown with (+).

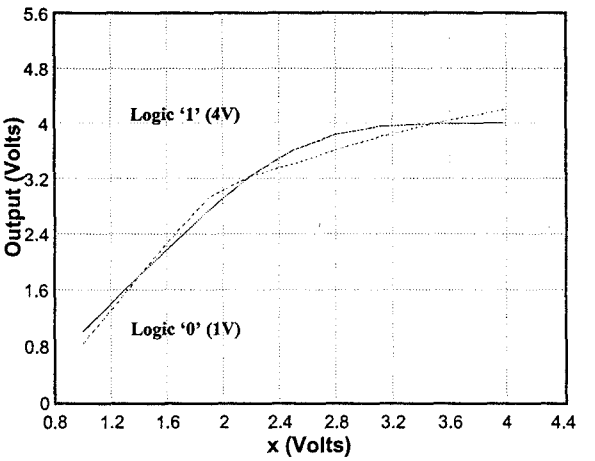


Figure 11. Diagonal cut for the response for the circuit implementing the fundamental S-norm for s=100. Full line depicts target characteristics.

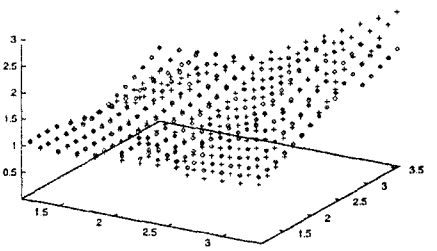

Figure 12. Measured response of a hardware-evolved circuit implementing the fundamental T-norm for s=100 (◊). Target characteristics are shown with (+).

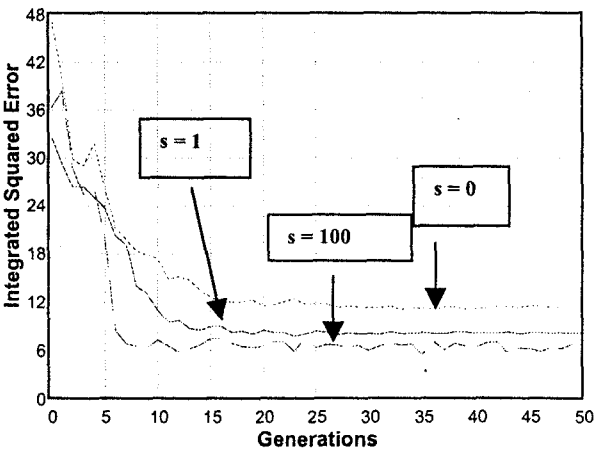


Figure 13. Decreasing error between best individual in each generation and target circuit, for the three software evolved circuits, with s=0, s=1, and s=100.

voltage value (◊ above +) than the ideal response for that input pair and sometimes it has a lower value (◊ below +). The errors are observed mainly at the domain extremes. Figure 13, where a function of the error of best individual is plotted across the number of generations, clearly shows decreasing error between best individual in each generation and target circuit, for the three software evolved circuits, with s=0, s=1, and s=100.

## 5. DISCUSSIONS:

The purpose of the results presented in this paper is to illustrate what can be obtained in a rapid evolution, with no prior knowledge on the circuit solution, with no optimization in terms of width (W) and length (L) of transistor channels, with limited resources (only those found in two FPTA cells). One limitation is the approximation error, ranging from 3.6% to a maximum of 9% MAPE (Mean Absolute Percent Error) in software and to a peak of 11.6% in hardware. Several factors can contribute to reducing the approximation error. One of them is to allow more flexibility in the selection of the points where the inputs are applied, and where the output is collected from. In this experiment, these were considered predetermined, however it is possible to let evolution decide where to interface the circuit with the input/output.

Another way to increase the approximation power is to allow more resources, e.g. allow resources from more than 2 cells. This is similar to increasing the approximation power of neural networks when extra neurons are added. The described experiments do not have any parametric adjustment. The width and length of the transistor channel were considered fixed. However previous results indicate that parametric optimization can produce good adjustments

after the topology has been determined [29]. This will also be possible in hardware since the new version of the chip will allow switch-selectable transistors with different W/L in the same cell [30].

These results are preliminary and are presented mainly to illustrate some aspects of the application of EHW to synthesis of electronic circuits implementing combinatorial fuzzy logic functions. No comparison with any state-of-the-art design tools is made, and, of course, the performance of (computer-assisted) human solutions could exceed the performance of the totally automated solutions illustrated here. However, to the best of our knowledge, complete automated design of the type presented here is not available with any other tool. Moreover, we believe that completely automated techniques of the kind presented here will surpass current design techniques within near future. The role of the humans would shift toward providing specifications and evolutionary pressures to guide the design to the desired result (which is not a trivial task).

## 6. CONCLUSIONS:

This paper presented our research in soft computing with dual focus of enabling architectures and high-speed parallel processing hardware. Results clearly demonstrate the orders of magnitude speed advantage with power-miser chip designs, predominantly in analog. In addition, an effort toward building evolution-oriented devices and tests have demonstrated how electronic circuits can be automatically synthesized, on the chip, to produce a desired circuit functionality. It illustrated the aspects of using evolvable hardware for the design of unconventional circuits such as combinatorial circuits for fuzzy logic.

## Acknowledgements:

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology and was jointly sponsored by the National Aeronautics and Space Administration and the Defense Advanced Research Projects Agency.

## References:

[1] A. Moopenn, T. Duong, and A.P. Thakoor, "Digital-analog hybrid synapse chips for electronic neural networks," Advances In: Neural Information Processing Systems –2 (NIPS-2); Ed: Touretzky 1990, Morgan Kaufmann, pp. 769-776.

[2] T. Duong, S.P. Eberhardt, M. Tran, T. Daud, A.P. Thakoor, "Learning and optimization with cascaded VLSI neural network building block chips, "Proc. IEEE/INNS International Joint Conference on Neural Networks (IJCNN), vol. I, 1992, pp. 184-189.

[3] S. Eberhardt, T. Duong, and A.P. Thakoor, "Considerations for hardware implementation of neural networks, "Proc. 22nd Asilomar Conf. On Signals, Systems, and Computers, vol. II, 1988, pp. 649-653.

[4] T. Daud, T. Duong, M. Tran, H. Langenbacher, and A. Thakoor, "High resolution synaptic weights and hardware-in-the-loop learning," Proc. SPIE Conf. On Nonlinear Image Processing VI: Ed: Dougherty, et al., vol. 2424, 1995, pp. 489-500.

[5] T. Duong, S. Kemeny, M. Tran, T. Daud, and A. Thakoor, "High speed low power analog ASICs for a 3-D neuroprocessor," Proc. SPIE Conf. On Nonlinear Image Processing VI; Ed: Dougherty, et al., vol. 2424, 1995, pp. 470-477.

[6] T. Duong, T. Daud, A. Thakoor, "On-chip learning of hyper-spectral data for real time target recognition," IASTED Conference on Intelligent Systems and Control (ISC), Honolulu, Hawaii, August 14-17, 2000 [to be published].

[7] A. Stoica, T. Thomas, W-T. Li, D.A. Weldon, T. Duong, and T. Daud, "Reconfigurable sensor fusion processing hardware," In: Intelligent Engineering Systems Through Artificial Neural Networks; Ed: C.H. Dagli, et al., ASME Press, New York, 1999, vol. 9, pp. 423-428.

[8] A. Stoica, D. Keymeulen, R. Tawel, C. Lazaro and Wei-te Li. "Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits." Proc. of the First NASA/DoD Workshop on Evolvable Hardware; Ed: A. Stoica, et al., July 19-21, 1999, IEEE Computer Society Press, pp. 77-84.

[9] D. Hammerstrom, E. Means, M. Griffin, G. Tahara, K. Knopp, R. Pinkham, and B. Riley, "An 11 million transistor digital neural network execution engine," *Proceedings of the IEEE International Solid-State Circuits Conference,* pp. 180-181, 1991.

[10] B. V. K. Vijaya Kumar, "Tutorial survey of composite filter designs for optical correlators," *Appl. Opt.*, vol. 31, no. 23, pp. 4773-4801, 1992.

[11] S. Udomkesmalee, A. Thakoor, C. Padgett, T. Daud, W-C. Fang, and S.C. Suddarth, "VIGILANTE: An advanced sensing/processing testbed for ATR applications," Proc. of the SPIE Conf. Automatic Target Recognition VII; Ed: F.A. Sadjadi, vol. 3069, 1997, pp. 82-93.

[12] J. Carson, "On focal plane array feature extraction using a 3-D artificial neural network (3DANN)," *Proc. SPIE*, vol. 1541, Part I: pp. 141-144, Part II: pp. 227-231, 1991.

[13] T. Duong, S. Kemeny, T. Daud, A. Thakoor, C. Saunders, and J. Carson, "Analog 3-D neuroprocessor for fast frame focal plane image processing," *SIMULATION*, vol. 65, no. 1, pp. 11-24, 1995.

[14] T. Duong, S. Kemeny, M. Tran, T. Daud, and A. Thakoor, "Low power analog neurosynapse chips for a 3-D sugarcube neuroprocessor," *Proceedings of the IEEE International Conference on Neural Networks*, vol. III, Orlando, pp. 1907-1911, 1994.

[15] T. Duong, T. Daud, and A.P. Thakoor, "Cascaded VLSI neural network architecture for on-line learning," U.S.A. Patent #5,479,579, Dec. 26, 1995.

[16] T. Daud, A. Stoica, T. Thomas, W-T. Li, and J. Fabunmi, "ELIPS: Toward a sensor fusion processor on a chip," SPIE Conf. Sensor Fusion: Architectures, Algorithms, and Applications III; Ed: B.V. Dasarathy, vol. 3719, 1999, pp. 209-219.

[17] T.J. Thomas and D.A. Weldon, "Digitally programmable current-mode analog fuzzy membership function circuit," In: Intelligent Engineering Systems Through Artificial Neural Networks; Ed: C.H. Dagli, et al., ASME Press, New York, 1999, vol. 9, pp. 623-628.

[18] R. Zebulum, A. Stoica and D. Keymeulen, "A flexible model of a CMOS field programmable transistor array targeted for hardware evolution", Third Int. Conference on Evolvable Systems: From Biology to Hardware (ICES2000), Edinburgh, April 17-19, 2000, to appear.

[19] A. Stoica, R. Zebulum and D. Keymeulen, "Mixtrinsic evolution," In: T. Fogarty, J. Miller, A. Thompson and P. Thompson, (eds.), Proceedings of the Third International Conference on Evolvable systems: From Biology to Hardware (ICES2000), April 17-19, 2000, Edinburgh, UK., New York, USA, Springer Verlag, to appear.

[20] J. Koza, F.H. Bennett, D. Andre, and M.A Keane, "Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming," *Proceedings of Genetic Programming Conference*, Stanford, CA, 1996, 28-31.

[21] A. Thompson, "An evolved circuit, intrinsic in silicon, entwined in physics". In *International Conference on Evolvable Systems*. Springer-Verlag Lecture Notes in Computer Science, 1996, 390-405.

[22] M. Sipper, D. Mange, A. Perez-Uribe (Eds.) "Evolvable systems: From biology to hardware," *Proc. of the Second International Conference, ICES 98*, Lausanne, Switzerland, Springer-Verlag, Lecture Notes in Computer Science, 1998.

[23] A. Stoica, D. Keymeulen, and J. Lohn (Eds.) *Proc. of the First NASA/DoD Workshop on Evolvable Hardware*, July 19-21, 1999, Pasadena, CA IEEE Computer Society Press.

[24] A.Stoica, "Toward evolvable hardware chips: experiments with a programmable transistor array." Proceedings of 7th International Conference on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, Granada, Spain, April 7-9, IEEE Comp Sci. Press, 1999, 156-162.

[25] Butnariu, D. and Klement, E. P., "Triangular Norm-Based Measures and Games with Fuzzy Coalitions," Kluwer Academics, 1993.

[26] M. M. Gupta and J. Qi, "Design of fuzzy logic controllers based on generalized t-operators", Fuzzy Sets and Systems, Vol. 40, 1991, 473-389.

[27] M.M. Gupta and J. Qi, "Theory of T-norms and fuzzy inference methods", Fuzzy Sets and Systems, Vol. 40, 1991, 431-450.

[28] Stoica, A., "Synaptic and somatic operators for fuzzy neurons: which T-norms to choose?" In Proc. of 1996 Biennial Conference of the North American Fuzzy Information Processing Society - NAFIPS, Berkeley, CA, June 19-22, 55-58.

[29] Stoica, A. "On hardware evolvability and levels of granularity", Proc. of the International Conference "Intelligent Systems and Semiotics 97: A Learning Perspective, NIST, Gaithersburg, MD, Sept. 22-25, 1997.

[30] A. Stoica, D. Keymeulen, V. Duong and C. Lazaro, "Automatic synthesis and fault-tolerant experiments on an evolvable hardware platform", In R. Profet et al.(Eds.), Proc. of IEEE Aerospace Conf., March 18-25, 2000, Big Sky, MO, IEEE Press.